Policy Management in CIFTS

PMan Policy Manager

Harish Naik
Mathematics & Computer Science Division
Argonne National Laboratory

December 1, 2010

Outline

Introduction

- More than one FTB enabled component may need to handle an event at the same time
- Typical scenarios
    1. Only one component should be allowed to handle event at one instance
    2. More than one can handle, but in a particular order
- A policy is necessary to resolve conflicts and define component behaviors

Principle

Consists of:

- Centralized system that has knowledge of all the policies defined for a particular system
- A policy definition file which has rules set for each event
- Policy priorities attributes for resolution of conflicts

Policy Management Process mainly consists of two phases:

1. Definition
2. Implementation

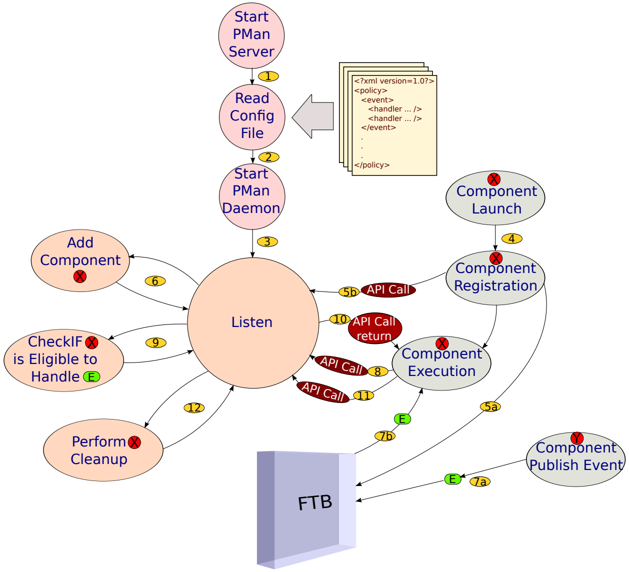
Policy Definition

- Component developers identify lists L_P & L_S
  - $L_P \leftarrow$ List of publishable events
  - $L_S \leftarrow$ List of subscribed events (Events that the component would like to respond to)
- L_S is more relevant from a policy handling perspective
- L_S is handed over to the system maintainer before component deployment
- System maintainer performs consolidation of all L_S of all the components on the system
- Policy aggregation & consolidation involves resolving conflicts among components (with common subscribed events) and assigning them priorities

Policy Implementation

- Every component identifies itself with the policy manager using a designated string. For example, all MPICH2 components would identify themselves with *MPICH2* string.
- The identifier string is helpful in matching the component to event handling entities in the policy definition (an XML file)
- When a component receives an event, it needs to call *pman_get_permission()* to check if the policy allows it to take action on the event
- The policy manager informs of its decision, based on the policy defined and the other components available on the system at that time
- Component takes action based on the permission
- Before exiting, the component calls *pman_unregister()* to inform the system that it will no longer participate
- Policy manager performs cleanup actions

Mechanism


Start PMan Server
1
Read Config File
<?xml version=1.0?>
<policy>
<event>
<handler ... />
<handler ... />
</event>
.
.
.
</policy>
2
Start PMan Daemon
3
Listen
Add Component X
6
CheckIF X is Eligible to Handle E
9
Perform X Cleanup
12
Component Launch X
4
Component Registration X
5b API Call
10 API Call return
Component Execution X
API Call 8
API Call 11
5a
E 7b
Component Publish Event Y
E 7a
FTB

Mechanism: Steps

1. PMan server starts with appropriate arguments
2. Loads policy file from the given XML Policy document
3. Starts PMan Daemon
4. FTB enabled component **X** launched (Component **Y** is also launched at some point)
5. Registration
    5. Connects and registers with FTB and subscribes to event **E**
    5. Registers with PMan with its specified standard string using *pman_register()*
6. PMan launches thread to add new component to its internal table

Mechanism: Steps

1. FTB Event publish & delivery
    1. FTB enabled component **Y** publishes event **E**
    1. FTB delivers **E** to component **X**
2. **X** asks permission to respond to event **E** by making the *pman_get_permission()* call
3. PMan thread checks if **X** has permission to handle **E** based on policy
4. PMan informs (*pman_get_permission()* returns) **X** of the decision, **X** takes action accordingly
5. Component **X** calls unregister using *pman_unregister()*
6. PMan unregisters **X** and performs cleanup

Limitations & TODOs

- Centralized scheme is not scalable. Need a distributed system with policy knowledge replicated.
- Only one component can take action at the moment. For scenarios where action by more than one component is allowed, we need to queue *pman_get_permission()* requests.
- Queued requests will be serviced as the current component reports completion of event handling
- Standardizing ID strings for different component types (eg. *MPICH2, COBALT, PVFS*)

More Information

For more information and updates go to:

http://wiki.mcs.anl.gov/cifts/index.php/Second_Approach_to_PM